Cuestionario POO 2020 preguntas y respuestas

Sistemas de Información (Universidad Abierta Interamericana)

PROGRAMACIÓN ORIENTADA A OBJETOS

CÓDIGO DE MATERIA 11
Titular: Darío Guillermo Cardacci

2020

ALUMNO: Agustín Porto Basavilbaso
Profesor: Leonardo Ghigliani

INSTITUTO SUPERIOR DEL PROFESORADO "JUAN B. ALBERDI" A-815

UNIVERSIDAD ABIERTA INTERAMERICANA

UNIDAD 1

1. *Enumere y explique los aspectos más relevantes que hacen que un software de gran magnitud sea complejo.*

Los sistemas de software complejos tienen un ciclo de vida largo y a lo largo del tiempo muchos usuarios llegan a depender de su funcionamiento correcto.
Resulta imposible para el desarrollador comprender todas las sutilidades de su diseño.
La complejidad de estos sistemas excede la capacidad intelectual humana. Estos sistemas son jerárquicos y los niveles de esta jerarquía representan diferentes niveles de abstracción, cada uno de los cuales se construye sobre el otro, y cada uno de los cuales es comprensible por sí mismo.

2. *¿Cuáles son los cinco atributos de un sistema complejo?*

Los atributos de los sistemas complejos son:

Tienen jerarquías: un sistema complejo se compone de subsistemas relacionados que tienen a su vez sus propios subsistemas y así sucesivamente hasta que se alcanza algún nivel ínfimo de componentes elementales

Componentes primitivos: La elección de qué componentes de un sistema son primitivos es relativamente arbitraria y queda en gran medida a decisión del observador.

Relacionamiento: Los enlaces internos de los componentes suelen ser más fuerte que los enlaces entre componentes. Esta diferencia entre interrelaciones intra-componentes e inter-componentes proporciona una separación clara de intereses entre las diferentes partes de un sistema, posibilitando el estudio de cada parte en forma relativamente aislada.

Tienen patrones comunes: Los sistemas jerárquicos están compuestos usualmente de solo unas pocas clases diferentes de sub-sistemas en varias combinaciones y disposiciones. Tienen patrones comunes que pueden conllevar a la reutilización de componentes pequeños.

Evolución: Un sistema complejo que funciona ha evolucionado en un sistema simple que funcionaba. Un sistema complejo diseñado desde cero nunca funciona y no puede parchearse para conseguir que lo haga. Hay que volver a empezar partiendo de un sistema simple que funcione. A medida que los sistemas evolucionan, objetos que en su momento se consideraban complejos se convierten en primitivos sobre los que se construyen sistemas más complejos aún.

3.
¿Cuáles son las dos jerarquías más importantes que consideramos en la Orientación a objetos para sistemas complejos?

- Estructura de Clases (es-un): Jerarquía de generalización/especialización. Se conoce como herencia. Define una relación entre clases, en donde una clase comparta la estructura o comportamiento definido en una o más clases. Herencia simple y múltiple.
- Estructura de objetos (parte-de): Jerarquía de agregación. Es el concepto que permite el agrupamiento físico de estructuras relacionadas lógicamente. Así un camión se compone de ruedas, motor, sistema de transmisión y chasis. En consecuencia, camión es una agregación y ruedas es el agregado de camión

4.
¿Con qué podemos enfrentar a la complejidad para obtener partes cada vez más pequeñas y simplificadas del dominio del problema?

La técnica de dominar la complejidad se conoce como divide y vencerás. Cuando se diseña un sistema de software complejo es esencial descomponerlo en partes más y más pequeñas, cada una de las cuales se puede refinar entonces de forma independiente. Para entender un nivel dado de un sistema basta con comprender unas pocas partes a la vez.

5. ¿Cuáles son las dos formas de descomposición más conocidas?

Las dos formas de descomposición son la algorítmica y la orientada a objetos.

6.
¿Explique en qué se diferencia la descomposición algorítmica y la orientada a objetos?

- Descomposición algorítmica: Análisis y diseño estructurado. Se aplica para descomponer un gran problema en pequeños problemas, la unidad fundamental de este tipo de descomposición es el subprograma. El programa resultante toma la forma de un árbol en el que cada subprograma realiza su trabajo, llamando ocasionalmente a otro subprograma
- Descomposición orientada a objetos: análisis y diseño orientado a objetos. El mundo es visto como un conjunto de entidades autónomas. Se descompone el sistema de acuerdo con las abstracciones clave del dominio del problema

Diferencia: la visión algorítmica enfatiza el orden de los eventos, la visión orientada a objetos resalta los agentes que, o bien causan acciones o bien son sujetos de esas acciones.

7. ¿Qué rol cumple la abstracción en la orientación a objetos?

Cuando se adopta una visión del mundo orientada a objetos, se toman a los objetos como abstracciones de entidades del mundo real. Representan un agrupamiento de información particularmente denso y cohesivo.
A través de la abstracción se utilizan bloques de información de contenido semántico cada vez mayor.

Incapaces es de dominar en su totalidad un objeto, decidimos ignorar sus detalles no esenciales, tratando en su lugar con el modelo generalizado e idealizado de objeto.

8. ¿Qué rol cumple la jerarquía en la orientación a objetos?

Otra forma de incrementar el contenido semántico de bloques de información individuales es mediante el reconocimiento explícito de las jerarquías de clases y objetos dentro de un sistema de software complejo.

La identificación de las jerarquías requiere que se descubran patrones entre muchos objetos, cada uno de los cuales puede incluir algún comportamiento tremendamente complicado. Una vez que se han expuesto estas jerarquías, la estructura de un sistema complejo experimenta una gran simplificación.

9. ¿Consideraría Ud. al diseño orientado a objetos un desarrollo evolutivo o revolucionario? Justifique.

Considero que es un desarrollo revolucionario porque cambia radicalmente la forma de pensar un sistema

10. ¿Cuántos y cuáles son los modelos básicos que se manejan en el diseño orientado a objetos?

El diseño orientado a objetos ofrece un rico conjunto de modelos que reflejan la importancia de plasmar explícitamente las jerarquías de clases y de objetos del sistema que se diseña:

Modelo estático: las propiedades de un objeto
Modelo dinámico: el valor de esas propiedades
Modelo lógico: estructura de clases y objetos
Modelo físico: arquitectura de módulos y procesos

11. *¿Qué es la programación orientada a objetos?*

La programación orientada a objetos es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase y cuyas clases son, todas ellas, miembros de una jerarquía de clases unidas mediante relaciones de herencia.

La programación orientada a objetos utiliza objetos, no algoritmos, como sus bloques lógicos de construcción fundamentales (jerarquía "parte de"), cada objeto es una instancia de alguna clase, las clases están relacionadas con otras clases por medio de relaciones de herencia (jerarquía de clases).

Si falta alguno de estos elementos un programa no es orientado a objetos.

12. *¿Qué es el diseño orientado a objetos?*

El diseño orientado a objetos es un método de diseño que abarca el proceso de descomposición orientada a objetos y una notación para describir los modelos lógico y físico, así como los modelos estático y dinámico del sistema que se diseña.

Modelos Lógicos (estructura de clases y objetos)

Modelos Físicos (arquitectura de módulos y procesos)

13. *¿Qué es el análisis orientado a objetos?*

El análisis orientado a objetos es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema

14. *¿Cuáles son los elementos fundamentales en el modelo de objetos?*

Abstracción

Encapsulamiento*

Modularidad

Jerarquía

15. *¿Cuáles son los elementos secundarios del modelo de objetos?*

Tipos

Concurrencia

Persistencia

16. *Explique el significado de la abstracción.*

Es una descripción simplificada o especificación de un sistema que enfatiza algunos de los detalles o propiedades del mismo mientras suprime otros.
Una abstracción denota las características esenciales de un objeto que lo distingue de todos los demás tipos de objeto y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador.

17. *Explique el significado del encapsulamiento.*

El encapsulamiento es el proceso de almacenar en un mismo comportamiento los elementos de una abstracción que constituyen su estructura y su comportamiento.
Sirve para separar la interface contractual de una abstracción y su implantación.

Se centra en la implementación que da lugar al comportamiento. Se consigue mediante la ocultación de información, que es el proceso de ocultar todas las propiedades de un objeto que no contribuyen a sus características esenciales. La estructura de un objeto está oculta así como la implantación de sus métodos.

18. *Explique el significado de la modularidad*.

La modularidad consiste en dividir un programa en módulos que pueden compilarse por separado, pero que tienen conexiones con otros módulos.

La modularidad es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados.
Agrupa abstracciones relacionadas lógicamente.

19. *Explique el significado de la jerarquía.*

La jerarquía es una clasificación u ordenación de abstracciones.
Las dos jerarquías más importantes en un sistema complejo son:
Su estructura de clases ("de clases") y su estructura de objetos ("de partes")

20. *Explique el significado de la tipificación.*

Un tipo es una caracterización precisa de propiedades estructurales o de comportamiento que comparten una serie de entidades.

Los tipos son la puesta en vigor de la clase de los objetos, de modo que los objetos de distintos tipos no pueden intercambiarse o como mucho pueden intercambiarse solo de formas muy restringidas.

21. *Explique el significado de la concurrencia.*

La concurrencia es la propiedad que distingue un objeto activo de uno que no está activo. Un sistema automatizado puede tener que manejar muchos eventos diferentes simultáneamente. Otros problemas pueden implicar tantos cálculos que excedan la capacidad de cualquier procesador individual. En ambos casos es natural considerar el uso de un conjunto distribuido de computadoras para la implantación que persigue utilizar procesadores de realizar multitareas.
La concurrencia permite a diferentes objetos actuar al mismo tiempo.

22. *Explique el significado de la persistencia.*

"La persistencia es la propiedad de un objeto por la que su existencia trasciende el tiempo (es decir, el objeto continúa existiendo después de que su creador deja de existir) y/o el espacio (es decir, la posición del objeto varía con respecto al espacio de direcciones en el que fue creado)". El tiempo de vida del objeto trasciende al tiempo de vida del programa que los creó.

La persistencia conserva el estado de un objeto en el tiempo y en el espacio.

La persistencia abarca algo más que la mera duración de los datos. En las BD orientadas a objetos, no sólo persiste el estado de un objeto, sino que su clase debe trascender también a cualquier programa individual, de forma que todos los programas interpreten de la misma manera el estado almacenado.

23. *¿Cómo se denotan las características esenciales de un objeto que lo distinguen de todos los demás tipos de objetos y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador?*

En términos generales, se define un objeto como cualquier cosa que tenga una frontera definida con nitidez. Pero esto no es suficiente para servir de guía al

distinguir un objeto de otro, ni permite juzgar la calidad de las abstracciones. Por lo tanto surge la siguiente definición:

Un objeto tiene estado, comportamiento e identidad; la estructura y comportamiento de objetos similares están definidos en su clase común, los términos instancia y objeto son intercambiables. El **estado** de un objeto abarca todas las propiedades del mismo (normalmente estáticas) más los valores actuales (normalmente dinámicos) de cada una de esas propiedades.

Una propiedad es una característica inherente o distintiva, un rasgo o cualidad que contribuye a hacer que un objeto sea ese objeto y no otro.
Las propiedades suelen ser estáticas, porque atributos como estos son inmutables y fundamentales para la naturaleza del objeto.
El hecho de que todo objeto tiene estado implica que todo objeto toma cierta cantidad de espacio.
El **comportamiento** es como actúa y reacciona un objeto en términos de sus cambios de estado y paso de mensajes. Representa su actividad visible y comprobable exteriormente.

24. *¿A qué denominamos un objeto cliente?*

Un cliente es cualquier objeto que utiliza los recursos de otro objeto (denominado servidor)

25. *¿A qué denominamos un objeto servidor?*

Un servidor es el que otorga recursos al objeto sin consumirlos.

26. *¿A qué denomina Meyer el modelo contractual de programación?*

Meyer denomina modelo contractual de programación a al contrato definido por la vista exterior de cada objeto que debe ser llevado a cabo por la vista interior del mismo y a su vez pueden depender de otros objetos.

27. *¿Qué establece un contrato entre objetos?*

Un contrato entre objetos establece todas las suposiciones que puede hacer un objeto cliente acerca del comportamiento de un objeto servidor. Abarca las responsabilidades de un objeto, es decir el comportamiento del que se le considera responsable.

28. *¿Cómo se denomina a las formas en que un objeto puede actuar y/o reaccionar, constituyendo estas formas la visión externa completa, estática y dinámica de la abstracción?*

A las formas en que un objeto puede actuar y/o reaccionar se lo denomina comportamiento.

29. *¿Cómo se denomina al conjunto completo de operaciones que puede realizar un cliente sobre un objeto, junto con las formas de invocación u órdenes que admite?*

Al conjunto completo de operaciones que puede realizar un cliente sobre un objeto, junto con las formas de invocación y órdenes que admite se le denomina protocolo.

30. *¿A qué nos referimos cuando decimos que un concepto central de la idea de abstracción es el de invariancia?*

Un invariante es una condición booleana (verdadera o falsa) cuyo valor de verdad debe mantenerse. La violación de un invariante rompe el contrato asociado con una abstracción.

31. *¿Qué se debe definir para cualquier operación asociada a un objeto?*

Para cualquier operación asociada con un objeto, es necesario definir precondiciones (invariantes asumidos por la operación) y pos condiciones (invariantes satisfechos por la operación)

32. *¿Qué es una precondición?*

Es una invariante asumida por una operación.

33. *¿Qué es una postcondición?*

Es una invariante satisfecha por una operación.

34. *¿A qué se denomina excepción?*

Una excepción es una indicación de que no se ha satisfecho (o no puede satisfacerse) alguna invariante. Algunos lenguajes permiten a los objetos lanzar excepciones, así como abandonar el procesamiento, avisando del problema a algún otro objeto que puede a su vez aceptar la excepción y tratar dicho problema.

35. *¿A qué se denomina mensaje?*

Un mensaje es una operación que un objeto realiza sobre otro.

36. *¿El encapsulado es un concepto complementario a la abstracción? Justifique.*

El encapsulado es un concepto complementario a la abstracción. Es el proceso de almacenar en un mismo compartimento los elementos de una abstracción que constituyen su estructura y su comportamiento. Sirve para separar el interfaz contractual de una abstracción y su implantación.

37. *¿Cómo se denomina al elemento de un objeto que captura su vista externa*

El elemento de un objeto que captura su vista externa se llama interfaz. La interfaz representa todo lo que un cliente necesita saber de una clase.

38.

¿Cómo se denomina al elemento de un objeto que captura su vista interna la cu al incluye los mecanismos que consiguen el comportamiento deseado?

Se denomina implementación.

39. *¿El concepto de "ocultar los detalles de implementación" lo asociaría a "esconder los detalles de implementación" o a "evitar el uso inapropiado de los detalles de implementación"? Justifique.*

Dicho concepto lo asociaría a ambas frases. Lo principal es evitar el uso inapropiado de detallas de implementación, como consecuencia de perseguir ese objetivo se esconden los detalles de implementación.

40. *¿Cuáles son los dos aspectos que hacen importante considerar a la modularidad?*

La modularidad es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados. Por lo tanto los dos aspectos que hacen importante considerar la modularidad son bajo acoplamiento y alta cohesión.

41. *¿Para qué se utiliza la jerarquía?*

Frecuentemente un conjunto de abstracciones forma una jerarquía, y la identificación de esas jerarquías en el diseño simplifica en gran medida la comprensión del problema.
Se define la jerarquía como una 3 de abstracciones.
Las dos jerarquías más importantes en un sistema complejo son su estructura de clases (la jerarquía "de clases") y su estructura de objetos (la jerarquía "de partes")

42.
¿Cómo denominamos a la caracterización precisa de propiedades estructurales o de comportamiento que comparten una serie de entidades?

Una caracterización precisa de propiedades estructurales o de comportamiento que comparten una serie de entidades se denomina tipo. Se utilizan los términos tipo y clase de manera intercambiable.
Los tipos son la puesta en vigor de la clase de los objetos, de modo que los objetos de tipos distintos no pueden intercambiarse o pueden intercambiarse de forma muy restringida.

43. *¿Las clases implementan tipos?*

Las clases contienen implementaciones de tipos

44. *¿Los tipos implementan clases?*

Los tipos no implementan clases

45. *¿Cómo denominamos a los lenguajes que hacen una comprobación de tipos estricta?*

Lenguajes fuertemente tipados

46.
¿Cómo denominamos a los lenguajes que no hacen una comprobación de tipos e stricta?

Lenguajes débilmente tipados

47. *¿A qué se denomina ligadura estática (temprana)?*

La asignación estática de tipos, también conocida como ligadura estática o ligadura temprana, se refiere al momento en el que los nombres se ligan con sus tipos. La ligadura estática significa que se fijan los tipos de todas las variables y expresiones en tiempo de compilación.

48. *¿A qué se denomina ligadura dinámica (tardía)?*

La ligadura dinámica, también llamada ligadura tardía, significa que los tipos de las variables y expresiones no se conocen hasta el tiempo de ejecución.

49. *¿Es lo mismo la comprobación estricta de tipos y la ligadura dinámica?*

No es lo mismo, son conceptos independientes. Un lenguaje puede tener comprobación estricta de tipos y tipos estáticos, puede tener comprobación estricta de tipos pero soportar enlace dinámico, o no tener tipos y admitir la ligadura dinámica.

50.
¿Cómo se denomina la característica que permite a diferentes objetos actuar al mismo tiempo?

Se denomina concurrencia

51. *¿A qué se denomina concurrencia pesada?*

Un proceso pesado es aquel típicamente manejado de forma independiente por el sistema operativo de destino y abarco su propio espacio de direcciones.

52. *¿A qué se denomina concurrencia ligera o liviana?*

Un proceso ligero suele existir dentro de un solo proceso del sistema operativo en compañía de otros procesos ligeros, que comparten el mismo espacio de direcciones.

53.
¿La concurrencia es la propiedad que distingue un objeto activo de uno que no l o está?

Si, la concurrencia es la propiedad que distingue un objeto activo de uno que no está activo.

54.
¿Cómo se denomina la característica en orientación a objetos que permite cons ervar el estado de un objeto en el tiempo y el espacio?

Se denomina persistencia.

55. *¿Qué cosas se persisten?*

Se persisten:

- Resultados transitorios en la evaluación de expresiones
- Variables locales en la activación de procedimientos
- Variables propias, variables globales y elementos del heap cuya duración difiere de su ámbito.
- Datos que existen entre ejecuciones de un programa
- Datos que existen entre varias versiones de un programa
- Datos que sobreviven al programa

56. *Defina qué es un objeto desde la perspectiva de la cognición humana.*

Desde la perspectiva de la cognición humana, un objeto es cualquiera de las siguientes cosas:

- Una cosa tangible y/o visible
- Algo que puede comprenderse intelectualmente
- Algo hacia lo que se dirige un pensamiento o acción

57. *¿Un objeto es real o abstracto? Justifique.*

Un objeto representa un elemento, unidad o entidad individual e identificable, ya sea real o abstracta, con un papel bien definido en el dominio del problema. Es cualquier cosa real o abstracta que posee una estructura que lo define y acciones que lo controlan.

58. *¿Los objetos poseen límites físicos precisos o imprecisos?*

Los objetos poseen limiten nítidamente definidos, es decir precisos.

59. *¿Cuáles son las tres cosas que debe tener un objeto?*

Para ser considerado objeto debe tener estado, comportamiento e identidad.

60. *¿Cuál es la palabra que se puede utilizar como sinónimo de objeto?*

Un sinónimo que se puede utilizar como objeto es instancia

61. *¿Cuál es la palabra que se puede utilizar como sinónimo de instancia?*

Objeto.

62. *¿Cómo definiría el estado de un objeto?*

Un objeto posee propiedades normalmente estáticas con valores normalmente dinámicos, el estado de un objeto es el valor que tiene dichas propiedades en un momento determinado

63. *¿A qué definimos propiedad de un objeto?*

Una propiedad es una característica inherente o distintiva, un rasgo o cualidad que contribuye a hacer que un objeto sea ese objeto y no otro. Las propiedades suelen ser estáticas, porque atributos como estos son inmutables y fundamentales para la naturaleza del objeto.

64. *¿Qué es lo que distingue a "un objeto" de los "valores simples"?*

Las cantidades simples son temporales, inmutables y no instanciadas, mientras que los objetos existen en el tiempo son modificables, tienen estado, son instanciados y pueden crearse, destruirse y compartirse

65. *¿Cómo definiría el comportamiento de un objeto?*

Ningún objeto existe de forma aislada. En vez de eso, los objetos reciben acciones y ellos mismos actúan sobre otros objetos. Así puede decirse que el comportamiento es cómo actúa y reacciona un objeto en términos de sus cambios de estado y paso de mensajes.

66.
¿El comportamiento de un objeto se ve afectado por el estado del mismo o bien que el comportamiento del objeto es función de su estado?

El estado de un objeto representa los resultados acumulados de su comportamiento.
El comportamiento de un objeto es función de su estado así como de la operación que se realiza sobre él, teniendo algunas operaciones efecto lateral de modificar el estado del objeto, entonces el estado de un objeto representa los resultados acumulados de su comportamiento.

67. ¿Algunos comportamientos pueden alterar el estado de un objeto?

Si

68.
Se puede afirmar que el estado de un objeto termina siendo los resultados acu mulados de su comportamiento.

Si, el estado de un objeto son los resultados acumulados de su comportamiento.

69. *¿A qué definiría como operación (método/función miembro)?*

Una operación es una acción que un objeto efectúa sobre otro con el fin de provocar una reacción. Las operaciones que los clientes pueden realizar sobre un objeto suelen declararse como métodos o función miembro.

70. *¿Cuáles son las tres operaciones más comunes?*

Una operación denota un servicio que una clase ofrece a sus clientes. Los tres tipos más comunes de operaciones son los siguientes:

- Modificador: Una operación que altera el estado de un objeto
- Selector: Una operación que accede al estado de un objeto pero no altera ese estado
- Iterador: Una operación que permite acceder a todas las partes de un objeto en algún orden perfectamente establecido.

71. ¿Cuáles son las dos operaciones habituales que se utilizan para crear y destruir instancias de clases?

- Constructor: Una operación que crea un objeto y/o inicializa su estado
- Destructor: Una operación que libera el estado de un objeto y/o destruye el propio objeto.

72. *¿Qué tipo de operación es el modificador?*

Una operación que altera el estado de un objeto

73. *¿Qué tipo de operación es el selector?*

Una operación que accede al estado de un objeto pero no altera ese estado

74. *¿Qué tipo de operación es el iterador?*

Una operación que permite acceder a todas las partes de un objeto en algún orden perfectamente establecido.

75. *¿Qué tipo de operación es el constructor?*

Una operación que crea un objeto y/o inicializa su estado

76. *¿Qué tipo de operación es el destructor?*

Una operación que libera el estado de un objeto y/o destruye el propio objeto.

77. ¿Cómo denominamos operaciones fuera de las clases en aquellos programas ori entados a objetos que permiten colocarlas (ej. C++)?

C++ permite escribir operaciones como subprogramas libres, llamados función no miembro. Los subprogramas libres son procedimientos o funciones que sirven como operaciones no primitivas sobre un objeto u objetos de la misma o de distintas clases.

78. *¿Todos los métodos son operaciones?*

Sí, todos los métodos son operaciones.

79. *¿Todas las operaciones son métodos?*

No. No todas las operaciones son métodos: algunas operaciones pueden expresarse como subprogramas libres.

80. *Dado el protocolo de un objeto (todos sus métodos y subprogramas libres asociados al objeto si el lenguaje lo permite) es conveniente dividirlo en grupos lógicos más pequeños de comportamiento? ¿Por qué?*

Para la mayoría de las abstracciones no triviales, es útil dividir este protocolo mayor en grupos lógicos de comportamiento. Estas colecciones, que constituyen una partición del espacio de comportamiento de un objeto, denotan los papeles que un objeto puede desempeñar y se definen los contratos entre las abstracciones y sus clientes.

81. *¿Cómo denominamos a los grupos lógicos más pequeños de comportamiento del protocolo total de un objeto?*

Papeles y responsabilidades

82. *¿Cuáles son las dos responsabilidades más importantes que posee un objeto?*

Las dos responsabilidades más importantes son: el conocimiento que un objeto mantiene y las acciones que puede llevar a cabo.

83. *¿Es relevante el orden en que se invocan las operaciones de un objeto?*

La existencia de un estado en el seno de un objeto significa que el orden en el que invocan las operaciones es importante. Para algunos objetos, esta ordenación de las operaciones respecto a los eventos y al tiempo es tan penetrante que se puede caracterizar mejor formalmente el comportamiento de tales objetos en términos de una máquina de estado finitos equivalente.

84. *¿Por qué decimos que los objetos se pueden considerar como máquinas?*

Las instancias de clases, actúan como pequeñas máquinas y, ya que todas esas instancias incorporan el mismo comportamiento, se puede utilizar la clase para capturar esta semántica común de orden respecto al tiempo y los eventos.

85. *¿Qué es la identidad de un objeto?*

La identidad es aquella propiedad de un objeto que lo distingue de todos los demás.

86. *Dadas dos variable X e Y del mismo tipo ¿qué significa que ambas son iguales?*

Que ambas apuntan a la misma dirección de memoria

87. *Dadas dos variable X e Y del mismo tipo ¿qué significa asignarle Y a X?*

Asignarle la dirección de memoria que apunta Y a X

88. *Dadas dos variable X e Y del mismo tipo ¿qué significa clonar X en Y?*

Copiar los valores de X en Y

89. *¿Qué significa realizar una clonación superficial?*

Significa que se copia el objeto pero no se comparte su estado

90. *¿Qué significa realizar una clonación profunda?*

Significa que se copia el objeto y su estado

91. *¿Qué es el ciclo de vida de un objeto?*

El ciclo de vida de un objeto es el tiempo de vida, el tiempo de vida de un objeto se extiende desde el momento que se crea por primera vez (y consume asi espacio) hasta que ese espacio se recupera

92. *¿Cómo se libera el espacio ocupado por un objeto?*

Los objetos creados en el heap con el operador new deben ser destruidos explícitamente con el operador GC.Collect.

93. *¿Qué tipos de relaciones existen entre los objetos?*

Existen dos tipos de relaciones:

- Enlaces
- Agregación

94. *¿Cómo podemos definir al enlace entre objetos?*

Enlace se define como una conexión física o conceptual entre objetos. Un objeto colabora con otros objetos a través de sus enlaces con estos. Un enlace denota la asociación especifica por la cual un objeto (cliente) utiliza los servicios de otro objeto (servidor), o a través del cual un objeto puede comunicarse con otro.

95. *Cómo pueden ser los mensajes entre dos objetos en una relación de enlace*

El paso de mensajes entre dos objetos es típicamente unidireccional aunque ocasionalmente puede ser bidireccional

96. *¿Qué es un mensaje unidireccional?*

Un objeto cliente invoca a un servidor

97. *¿Qué es un mensaje bidireccional?*

Los datos pueden fluir en ambas direcciones

98. *¿Quién inicia el paso de un mensaje entre dos objetos en una relación de enlace?*

El paso de mensajes es iniciado por el cliente, dirigido hacia el servidor

99. ¿Cuáles son los roles o papeles que puede desempeñar un objeto en una relación de enlace?

Los roles que puede desempeñar un objeto en una relación de enlace son: Actor, Servidor, Agente

100. *¿Qué significa que un objeto actúe como "Actor"?*

Un objeto que puede operar sobre otros objetos pero nunca se opera sobre él por parte de otros objetos, en algunos contextos, el término objeto activo y actor son equivalentes.

101. *¿Qué significa que un objeto actúe como "Servidor"?*

Un objeto que nunca opera sobre otro objetos, los otros objetos operan sobre el

102. *¿Qué significa que un objeto actúe como "Agente"?*

Un objeto que puede operar sobre otros objetos y además otros objetos pueden operar sobre él. Un agente se crea normalmente para realizar algún trabajo en nombre de un actor y otro agente

103. *Dados dos objetos A y B, si A le puede enviar un mensaje a B, estando ambos relacionados por enlace, decimos que B respecto de A está:*

Visible

104. ¿Cuáles son las cuatro formas de visibilidad que puede poseer un objeto servidor respecto de un objeto cliente?

- El objeto servidor es global para el cliente
- El objeto servidor es un parámetro de alguna operación del cliente
- El objeto servidor es parte del objeto cliente
- El objeto servidor es un objeto declarado localmente en alguna operación del cliente.

105. En una relación de enlace de dos objetos, cuando uno le pasa un mensaje al otro , además de adoptar roles ambos deben estar:

Sincronizados.

106. ¿Cuáles son las posibles formas de sincronización?

- Secuencial: La semántica del objeto pasivo está garantizada solo en presencia de un único objeto activo simultáneamente
- Vigilado: La semántica del objeto pasivo está garantizada en presencia de múltiples hilos de control, pero los clientes activos deben colaborar para lograr la exclusión mutua.
- Síncrono: La semántica del objeto pasivo está garantizada en presencia de múltiples hilos de control y el servidor garantiza la exclusión mutua.

107. ¿Qué significa que dados dos objetos A y B estos están secuencialmente sincronizados?

El funcionamiento del objeto pasivo está garantizado por el accionar de un único objeto activo simultáneamente. Hay un solo hilo de ejecución o de atención (uno solo por vez)

108. ¿Qué significa que la forma de sincronizarse de un conjunto de objetos es vigila da?

La semántica del objeto pasivo está garantizada en presencia de múltiples hilos de control, pero los clientes activos deben colaborar para lograr la exclusión mutua.

109. ¿Qué significa que la forma de sincronizarse de un conjunto de objetos es síncro na?

La semántica del objeto pasivo está garantizada en presencia de múltiples hilos de control y el servidor garantiza la exclusión mutua.

110. *¿El enlace es una relación de igual a igual o jerárquica?*

Los enlaces denotan relaciones de igual a igual o cliente servidor.

111. *¿La agregación es una relación de igual a igual o jerárquica?*

La agregación denota una jerarquía todo/parte, con la capacidad de ir desde el todo (también llamado el agregado) hasta sus partes (conocidas también como atributos)

112. *¿Qué tipo de jerarquía denota la agregación?*

Todo/parte

113. *¿Qué otros nombres recibe el "todo" en una relación de agregación?*

Agregado

114. *¿En una relación de agregación las "partes" forman parte del estado del "todo"?*

Si, son los atributos

115. *¿Qué tipos de agregación existen?*

- Con contención física: Cuando un objeto no existe sin el otro. Sus ciclos de vida están íntimamente relacionados. Por ejemplo, un avión se compone de alas, motores, etc (es un caso de contención física).

- Sin contención física: El ciclo de vida de los elementos son independientes uno del otro. Ejemplo: la relación entre un accionista y sus acciones es una relación de agregación que no requiere contención física. El accionista únicamente posee acciones, pero las mismas no son de ninguna manera parte física del accionista.

116. *¿Qué caracteriza a la agregación con contención física?*

Cuando un objeto no existe sin el otro. Sus ciclos de vida están íntimamente relacionados. Por ejemplo un avión se compone de alas, motores, etc (es un caso de contención física)

117. *¿Qué es una clase?*

Una clase representa solo una abstracción, la esencia de un objeto. Se puede definir una clase como un grupo conjunto o tipo marcado por atributos comunes o un atributo común; una división, distinción o clasificación de grupos basada en la calidad, grado de competencia o condición.

118. *¿La interfaz de la clase proporciona su visión interna?*

No esto lo proporciona la implementación, la implementación de una clase es su visión interna, que engloba los secretos de su comportamiento. La implementación de una clase se compone principalmente de la implementación de todas las operaciones definidas en la interfaz de esta.

119. *¿La implementación de la clase proporciona su visión externa?*

No, esto lo proporciona la interfaz, la interfaz de una clase proporciona su visión externa y por lo tanto enfatiza la abstracción a la vez que oculta su estructura y los secretos de su comportamiento. Este interfaz se compone principalmente de las declaraciones de todas las operaciones aplicables a instancias de esta clase, pero también puede incluir la declaración de otras clases, constantes, variables y excepciones, según se necesiten para completar la abstracción.

120.

¿En cuántas partes la podemos dividir una interfaz en términos de la accesibilid ad o visibilidad que posee?

Se puede dividir en 4 partes:

- Public: una declaración accesible a todos los clientes.
- Protected: una declaración accesible sólo a la propia clase, sus subclases, y sus clases amigas.
- Internal: Solo puede accederse dentro del proyecto
- Private: una declaración accesible sólo a la propia clase y sus clases amigas.

121. *¿Qué tipos básicos de relaciones existen entre las clases?*

Existen tres tipos básicos de relaciones entre clases:

- Generalización/especialización: denota una relación es un. Por ejemplo, una rosa es un tipo de flor, lo que quiere decir que una rosa es una subclase especializada de una clase más general, la de las flores.
- Todo/Parte: Denota una relación parte de, así un pétalo no es un tipo de flor, es parte de una flor.
- Asociación: Denota alguna dependencia semántica entre clases de otro modo independientes. Por ejemplo, las rosas y las velas son clases claramente independientes pero ambas representan cosas que podrían utilizarse para decorar la mesa de una cena.

122. *¿Qué relaciones entre clases se desprenden de las tres relaciones básicas?*

- Herencia,
- Agregación
- Uso
- Instanciación

123. *¿La asociación denota una dependencia semántica y la dirección de esta asociación?*

Una asociación solo denota una dependencia semántica y no establece la dirección de esta dependencia ni establece la forma exacta en que una clase se relaciona con otra.

La asociación puede ser bidireccional

124. *¿Qué significa la cardinalidad en una relación?*

Es la multiplicidad con que se relaciona una clase con otra.

125. *¿Qué cardinalidad puede existir entre clases relacionadas por asociación?*

- Uno a uno
- Uno a muchos
- Muchos a muchos

126. *¿Qué es la herencia?*

La herencia es una relación entre clases en la que una clase comparte la estructura y/o el comportamiento definidos en una o más clases. La clase de la que

las otras heredan se denomina superclase. La clase que hereda de otra se denomina subclase.

127. *¿Cuántos tipos de herencia existen?*

Existen dos tipos:

- Herencia Simple
- Herencia múltiple

128. *¿A qué se denomina herencia simple?*

Una clase hereda parte de la estructura y/o comportamiento de otra clase

129. *¿A qué se denomina herencia múltiple?*

Una clase hereda parte de la estructura y/o comportamiento de más de una clase

130. *¿Cómo se denomina a la clase que no se espera tener instancias de ella y solo se utilizará para heredar a otras clases?*

Se denomina clase abstracta.

131.¿Cómo se denomina a la clase que se espera tener instancias de ella y pued e utilizarse para heredar a otras clases o no?

Se denomina clase concreta.

132. *¿Cómo se denomina al método de una clase abstracta que no posee implementación y fue pensado para que sea implementado en las sub clases que lo heredan?*

Método virtual.

133. *¿Cómo se denomina a la clase más generalizada en una estructura de clases?*

superclase o clase base.

134. *¿Qué es el polimorfismo?*

El polimorfismo es un concepto de teoría de tipos en el que un nombre puede denotar instancias de muchas clases diferentes en tanto y en cuanto estén relacionadas por alguna superclase común. Cualquier objeto denotado por este nombre es, por lo tanto capaz de responder a algún conjunto común de operaciones de diversas formas.

135. *¿Cómo se denomina cuando una clase posee métodos que comparten el nombre y se diferencias por su firma?*

Se denomina sobrecarga de métodos.

136. *¿Qué sentencias de código se evitan utilizar cuando se aplica correctamente el polimorfismo?*

Se evita el uso de if anidados, que generan un código altamente acoplado.

137. *¿Qué es la agregación cómo relación entre clases?*

Las relaciones de agregación entre clases tienen un paralelismo directo con las relaciones de agregación entre los objetos correspondientes a esas clases. Relación Jerárquica del tipo Todo / Parte. La agregación es un caso particular de la asociación que denota posesión.

138. *¿Qué formas de contención física existen en la agregación?*

- Con contención Física : se subdivide en valor y referencia
- Sin contención Física: el ciclo de vida de los elementos son independientes uno del otro

139. *¿Qué características posee la contención física por valor?*

Contención física por valor: es un tipo de contención que significa que el objeto contenido no existe independientemente de la instancia contenedora que lo encierra. Por el contrario, el tiempo de vida de ambos objetos está en intima conexión. Cuando se crea una instancia del contenedor se crea una del contenido. Cuando se destruye el contenedor, se destruye el contenido.

140. *¿Qué características posee la contención física por referencia?*

Los tiempos de vida de ambos objetos ya no están tan estrechamente emparejados como antes. Se pude crear y destruir instancias de cada clase independientemente.

141. *¿Qué es una relación de uso?*

Una asociación denota una conexión semántica bidireccional, una relación de uso es un posible refinamiento de una asociación, por el que se establece qué abstracción es el cliente y qué abstracción es el servidor que proporciona ciertos servicios. Es una asociación refinada, se especifican roles (actor, cliente, servidor)

142. *¿Qué es la instanciación?*

La instanciación es la creación de instancias, es la acción por la cual se somete a una clase para obtener un objeto

143. *¿Todo objeto es una instancia de una clase?*

Sí, todo objeto es una instancia de una clase.

144. *¿Qué es una metaclase?*

Una metaclase es una clase cuyas instancias son clases.

145. *¿Qué métricas hay que observar para determinar la calidad de una abstracción?*

- Bajo acoplamiento
- Alta cohesión
- Suficiencia
- Compleción
- Ser primitivo

146. *¿Qué es el acoplamiento?*

La medida de la fuerza de la asociación establecida por una conexión entre un módulo y otro. El acoplamiento fuerte complica un sistema porque los módulos son más difíciles de comprender, cambiar o corregir por si mismos si están muy interrelacionados con otros módulos. La complejidad puede reducirse diseñando sistemas con los acoplamientos más débiles posibles entre los módulos.

147. *¿Qué es la cohesión?*

La cohesión mide el grado de conectividad entre los elementos de un solo modulo (y para el diseño orientado a objetos, una clase u objeto). La forma de cohesión menos deseable es la cohesión por coincidencia, en la que se incluyen en la misma clase o modulo abstracciones sin ninguna relación. La forma más deseable de cohesión es la cohesión funcional, en la cual los elementos de una clase o modulo trabajan todos juntos para proporcionar algún comportamiento bien delimitado.

148. *¿Qué es la suficiencia?*

La clase o módulo captura suficientes características de la abstracción como para permitir una interacción significativa y eficiente. Lo contrario produce componentes inútiles. En otras palabras es el grado en que una clase representa elementos suficientes y necesarios para que sea una implementación eficiente.

149. *¿Qué es la compleción?*

Por completo, quiere decirse que la interfaz de la clase o módulos captura todas las características significativas de la abstracción. Mientas la suficiencia implica una interfaz mínima, una interfaz completa es aquella que cubre todos los aspectos de la abstracción. Una clase o módulo completo es aquel cuya interfaz es suficientemente general para ser utilizable de forma común por cualquier cliente. Según apuntes de clase: es el grado en que la clase contempla características significativas de la abstracción.

150. *¿Qué significa ser primitivo?*

Las operaciones primitivas son aquellas que pueden implantarse eficientemente sólo si tienen acceso a la representación subyacente de la abstracción. Apunte de clase: lo más básica posible para que sea reutilizable. Todo esto depende del contexto.

151. *¿Qué se debe observar al momento de decidir si una abstracción debe implementar un determinado comportamiento o no?*

- Reutilización (Reusabilidad): ¿sería este comportamiento más útil en más de un contexto?
- Complejidad: ¿qué grado de dificultad plantea implementar este comportamiento?
- Aplicabilidad: ¿qué relevancia tiene este comportamiento para el tipo en el que podría ubicarse?
- Conocimiento de la implementación: ¿depende del comportamiento de los detalles internos?

152. *¿Qué formas puede adoptar el paso de un mensaje?*

- Síncrono
- Abandono inmediato
- De intervalo
- Asíncrono

153. *¿Qué características posee un mensaje síncrono?*

Una operación comienza sólo cuando el emisor ha iniciado la acción y el receptor está preparado para aceptar el mensaje, el emisor y el receptor esperarán indefinidamente hasta que ambas partes estén preparadas para continuar.

154. *¿Qué características posee un mensaje de abandono inmediato?*

Igual que el síncrono, excepto en que el emisor abandonará la operación si el receptor no está preparado inmediatamente.

155. *¿Qué características posee un mensaje de intervalo?*

Igual que el síncrono excepto en que el emisor esperará a que el receptor esté listo solo durante un intervalo de tiempo especificado.

156. *¿Qué características posee un mensaje Asíncrono?*

Un emisor puede iniciar una acción independientemente de si el receptor está esperando o no el mensaje.

157. *¿Qué significa que una abstracción está accesible a otra?*

Por accesible se entienda la capacidad de una abstracción para ver a otra y hacer referencia a recursos de su vista externa. Una abstracción es accesible a otro sólo donde sus ámbitos se superpongan y solo donde estén garantizados los derechos de acceso.

158. *¿Qué expresa la ley de Demeter?*

La ley Demeter afirma que los métodos de una clase no deberían depender de ninguna manera de la estructura de ninguna clase, salvo de la estructura inmediata (nivel superior) de su propia clase. Además, cada método debería enviar mensajes solo a objetos pertenecientes a un conjunto muy limitado de clases.

159. *¿Cuál es la consecuencia inmediata de aplicar la ley de Demeter?*

El efecto básico de la aplicación de esta ley es la creación de clases débilmente acopladas, cuyos secretos de implantación están encapsulados. Tales clases están claramente libres de sobrecargas, lo que significa que para comprender el significado de una clase no es necesario comprender los detalles de muchas otras clases.

160. *¿Cuáles son las cuatro formas fundamentales por las cuales un objeto X puede hacerse visible a un objeto Y?*

- El objeto proveedor es global al cliente.
- El objeto proveedor es parámetro de alguna operación del cliente.
- El objeto proveedor es parte del objeto cliente.
- El objeto proveedor es un objeto declarado localmente en el ámbito del diagrama de objetos.

161. *¿Para qué sirve clasificar a los objetos?*

La clasificación ayuda a identificar jerarquías de generalización, especialización y agregación entre clases. Reconociendo los patrones comunes de interacción entre objetos, se llega a idear mecanismos que sirven como alma de la implantación. La clasificación también proporciona una guía para tomar decisiones sobre modularización.

162. *¿Por qué es tan difícil la clasificación de objetos?*

Porque cualquier clasificación es relativa a la perspectiva del observador que la realiza (hay tantas formas de dividir el mundo en sistemas de objetos como científicos para emprender esa tarea) y la clasificación inteligente requiere una gran cantidad de imaginación creativa.

163. *¿Cómo es el rol del observador en la clasificación de objetos?*

El rol es muy importante. El observador debe asimilar la información derivada de sus observaciones y después sacar conclusiones acerca de sus construcciones hipotéticas. Puede hacer inferencias por completo erróneas. Por el contrario, si el observador es por completo objetivo y no conoce el tema de la observación puede que lo observado no sea lo adecuado. La observación exige un conocimiento competente de lo observado y de su significado.

164. *¿Cuáles son las aproximaciones generales a la clasificación?*

- Categorización Clásica
- Agrupamiento Conceptual
- Teoría de prototipos

165. *¿Qué es la categorización clásica?*

Todas las entidades que tienen una determinada propiedad o colección de propiedades en común forman una categoría. Tales propiedades son necesarias y suficientes para definir la categoría.

166. *¿Qué es el agrupamiento conceptual?*

En este enfoque las clases (agrupaciones de entidades), se generan formulando primero descripciones conceptuales de estas clases y clasificando entonces las entidades de acuerdo con las descripciones.

167. *¿Qué es la teoría de prototipos?*

Se crean clases prototípicas. A todas aquellas que se le aproximan en forma significativa se las considera pertenecientes a ese tipo. Parte de escoger un objeto prototipo que representa a una clase de objetos, y considerar a otros objetos como miembros de la clase si y sólo si se parecen de modo significativo al prototipo.

168. *¿Qué es una abstracción clave?*

Es una clase u objeto que forma parte del vocabulario del dominio del problema. El valor principal que tiene la identificación de tales abstracciones es que dan unos límites al problema; enfatizan las cosas que están en el sistema y, por tanto, son relevantes para el diseño, y suprimen las cosas que están fuera del sistema que son superfluas. La identificación de abstracciones clave es altamente específica de cada dominio.

Su identificación requiere descubrimiento e invención. Esta identificación inicial da lugar a abstracciones que, en muchos casos, resultan erróneas. Requieren un proceso de refinado y estructuración jerárquica. Se aconseja seguir las reglas de notación:

- Los objetos deben denominarse con nombres propios, mientras las clases tienen nombres comunes.
- Operaciones modificadoras deben denominarse con verbos activos, mientras las operaciones de selección usan preguntas o nombres con el verbo ser.

169. *¿Qué son los mecanismos?*

Mecanismo es cualquier estructura mediante la cual los objetos colaboran para proporcionar algún comportamiento que satisficiese un requerimiento del problema. Mientras el diseño de una clase incorpora el conocimiento de cómo se comportan los objetos individuales, un mecanismo es una decisión de diseño sobre como cooperan colecciones de objetos. Los mecanismos representan así patrones de comportamiento.

UNIDAD 2

170. *¿Qué es un Campo de una clase?*

El campo de una clase es una propiedad o atributo

171. *¿Qué es un método de una clase?*

Es un procedimiento o función que modifica el estado de una clase

172. *¿A qué denominamos sobrecarga?*

Sobrecargar un método significa que podrá proporcionar varios métodos con el mismo nombre pero con diferentes firmas de parámetros (Es decir, con un numero distinto de parámetros o con parámetros de diferentes tipos)

173. *¿Qué es una propiedad de una clase?*

Las propiedades de una clase son elementos que permiten acceder a atributos de la clase a través de métodos get y set.

174. *¿Qué tipos de propiedades existen?*

- Solo Lectura
- Solo Escritura
- Lectura/Escritura
- Propiedades con Argumentos

175. *¿Qué ámbitos pueden tener los campos, métodos y propiedades de las clases?*

- Private
- Protected
- Internal
- Public

176.¿Qué características posee cada ámbito existente si se lo aplico a un campo, un método y una propiedad de una clase?

- **Private:** Un miembro declarado como Privado, sólo es accesible en la clase en la que se ha declarado.

- **Protected:** Un miembro declarado como Protected es accesible en la clase en la que se ha declarado y en las que deriven de ella.

- **Internal:** Serán accesibles desde el ensamblado en el cual están declarados y tampoco se pueden utilizar en el interior de una función.

- **Public:** Visible por todo el mundo

177. *¿Para qué se utilizan los constructores?*

Es un método que se ejecutan cuando se crea una nueva instancia de la clase. En dicho método se inicializa el estado de un objeto y se crea la contención física para el mismo.

178. *¿A qué se denomina tiempo de vida de un objeto?*

El tiempo de vida es el ciclo durante el cual el objeto vive en el programa, abarca la creación modificación y destrucción del mismo.

179. *¿Para administrar las instancias de .NET se utiliza un contador de referencias?*

No se utiliza un contador de referencias.

180. *¿Qué objeto es el encargado de liberar el espacio ocupado por objetos que ya no se utilizan?*

El encargado de liberar el espacio ocupado por objetos que ya no se utilizan es el garbage collector.

181. *¿Qué son los sucesos?*

Los eventos o sucesos son el modo que tiene una clase de proporcionar notificaciones a sus clientes cuando ocurre una determinada accion.

182. *¿Qué se utiliza para declarar un suceso?*

Se utiliza la palabra reservada event.

183. *¿Cómo se logra que ocurra un suceso?*

184. *¿Cómo se pueden atrapar los sucesos?*

Con EventHandler

185. *¿Para qué se utiliza Addhandler en un suceso?*

Se utiliza para iniciar eventos en tiempo de ejecución.

186. *¿Cómo y que cosas se pueden compartir en una clase?*

Campos, métodos y sucesos.

187. *¿Qué característica poseen los campos compartidos?*

188. *Que características poseen los métodos compartidos?*

189. *¿Qué características poseen los sucesos compartidos?*

190. *¿Qué son y para que se pueden utilizar las clases anidadas?*

Las clases anidadas son clases dentro de otras. La característica principal de las clases anidadas es que pueden acceder a miembros privados de la clase externa mientras tienen todo el poder de una clase.

191. *¿Qué ámbitos/modificadores de acceso existen? Explique las características de cada uno.*

Los ámbitos que existen en C# son los siguientes:

- Public: Los elementos declarados serán accesibles desde cualquier porción de código del proyecto y desde cualquier otro proyecto que haga referencia a aquel donde están declarados.
- Protected: Se puede utilizar en el interior de una clase. Permite restringir el acceso a la variable únicamente al código de la clase y todas las clases que hereden de ella.
- Private: Restringe el acceso a la variable al módulo, a la clase o a la estructura en la cual está declarada. No se puede utilizar en el interior de un procedimiento o función.

192. *¿Qué cosas se pueden heredar?*

Se pueden heredar todos los componentes de una clase, menos los constructores estáticos, constructores de instancia y destructores.

193. *¿Cómo y para que se puede aprovechar en la practica el polimorfismo?*

Se utiliza para reimplementar métodos en clases hijas y así poder extender sus funciones. Se utiliza la palabra overide para reimplementar métodos virtuales y abstractos.

194. *¿Cómo y para que se utiliza la clase derived?*

Se denomina clase derivada a la clase que hereda los miembros de la clase base.

195. *¿Qué representa this?*

Sirve para referenciar la instancia actual de la clase donde se invoca.

196. *¿Qué clase representa a la clase base?*

Representa la superclase de la cual hereda la clase en la que se está posicionado

197. *¿Para qué se usa una clase abstracta?*

Es una clase que puede ser heredada pero no instanciada. Se usa para agrupar atributos y métodos que queramos tener en determinadas clases hijas pero que no sean instanciados.

198. *¿Para que se usa una clase sellada?*

Se usan cuando queremos generar clases que no tengan ningún tipo de herencia. Es un nivel superior de encapsulamiento.

199. *¿Qué es la sobreescritura?*

La sobreescritura es un mecanismo que usamos para sobreescribir métodos de la clase que estamos heredando, para modificar su uso internamente en otra clase.

200. *¿Qué elementos se pueden sobreescribir?*

Los métodos.

201. *¿A que se denomina sombreado de metodos?*

El sombreado de métodos se usa para ocultar métodos de una clase base.

202. *¿Qué característica peculiar posee el sombreado vs la sobre-escritura?*

UNIDAD 3

203. *¿Qué es un Framework?*

Es un marco de trabajo que ofrece a quien lo utiliza, una serie de herramientas para facilitarle la realización de tareas. Puede contener librerías de clases, documentación, ayuda, ejemplos, tutoriales pero sobre todo contiene experiencia sobre un dominio de problema específico. Resuelve cuestiones que son comunes a varios sistemas que tienen ciertas similitudes. Es una solución para un entorno de trabajo.

Características principales:

1. Normalizar: homogeneizar para usarlo en forma común y colocarlo en el framework
2. Estandarizar: generar documentación
3. Reutilizar experiencia

204. *¿Qué son los frozen-spots en un Framework?*

Algunas de las características del framework no son mutables ni tampoco pueden ser alteradas fácilmente. Estos puntos inmutables constituyen el núcleo o kernel de un framework, también llamados como los puntos congelados o frozen-spots del framework.
Los puntos congelados o inmutables son los pedacitos del código puestos en ejecución ya dentro del framework que llaman a uno o más puntos calientes

proporcionados por el ejecutor. El núcleo o Kernel será la constante y presentará siempre la parte de cada instancia del framework

205. *¿Qué son los hot-spots en un Framework?*

Los puntos calientes o Hot-spots son las clases o los métodos abstractos que deben ser implementados o puestos en ejecución. Es el servicio que presta el framework que requiere especificación de detalle a nivel de aplicación.

206. *¿Cómo se puede clasificar un Framework según su extensibilidad?*

Un framework se puede también clasificar según su extensibilidad; puede ser utilizado como una caja blanca o caja negra.

207. *¿Qué es un Framework de Caja Blanca?*

Las instantiaciónes son solamente posibles a través de la creación de nuevas clases. Estas clases y el código se pueden introducir en el marco por herencia o composición. Uno debe programar el framework y entenderlo muy bien para producir un caso.

208. *¿Qué es un Framework de Caja Negra?*

Los frameworks de caja negra producen instancias usando escrituras o scripts de configuración. Después de la configuración, una herramienta automática de instanciación crea las clases y el código de fuente (Source Code).

209. *¿Qué ventajas posee utilizar un Framework?*

- Permite abstraerse de problemas resueltos con amplia experiencia previa.
- En el caso de.NET el programador no piensa en arquitectura sino en programas.
- Permite aprovechar todas las ventajas de la estandarización.
- Las aplicaciones crecen y necesitan mantenimiento. Los frameworks facilitan la separación de la presentación y la lógica además de mantener una sintaxis coherente.

210. *¿Qué problemas resuelve .NET Framework?*

- Desde Internet, muchas aplicaciones y dispositivos están fuertemente comunicados entre sí.
- Los programadores escribían arquitectura en lugar de aplicaciones.
- Los programadores tenían conocimientos limitados o debían aprender nuevos lenguajes.
- El .NET Framework constituye las bases sobre las que, tanto aplicaciones como servicios, son ejecutadas y construidas.
- La naturaleza unificada del .NET Framework permite que cualquier tipo de aplicación sea desarrollada mediante herramientas comunes haciendo la integración mucho más simple.

211. *¿Qué es y qué permite hacer el CLR?*

El Common Language Runtime o CLR ("entorno en tiempo de ejecución de lenguaje común") es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET. El CLR es el encargado de compilar una forma de código intermedio llamada MSIL (Microsoft Intermediate Language), al código de maquina nativo, mediante un compilador en tiempo de ejecución.1No debe confundirse el CLR con una máquina virtual, ya que una vez que el código está compilado, corre nativamente sin intervención de una capa de abstracción sobre el hardware subyacente.

212. *¿Qué es el MSIL?*

A diferencia de lo que sucede en los lenguajes de programación tradicionales, los compiladores .NET no producen código nativo que pueda inyectarse directamente en la CPU y ser ejecutado por ésta. En su lugar, producen el denominado código en Lenguaje Intermedio de Microsoft (MSIL o IL) que es una especia de lenguaje máquina asociado con un procesador virtual que no se corresponde con ninguna CPU. Es de un nivel superior al puro lenguaje ensamblador, tiene en cuenta conceptos de alto nivel tales como excepciones y creaciones de objetos.

213. *¿Qué es el CTS?*

El Common Type System (CTS) o Sistema de Tipo Común es el conjunto de reglas que han de seguir las definiciones de tipos de datos para que el CLR las acepte. Es decir, aunque cada lenguaje gestionado disponga de su propia sintaxis para definir tipos de datos, en el MSIL resultante de la compilación de sus códigos fuente se han de cumplir las reglas del CTS. Algunos ejemplos de estas reglas son:

- Cada tipo de dato puede constar de cero o más miembros. Cada uno de estos miembros puede ser un campo, un método, una propiedad o un evento.

- No puede haber herencia múltiple, y todo tipo de dato ha de heredar directa o indirectamente deSystem.Object.

214. *¿Qué es el CLS?*

El CLS es la especificación de lenguaje común, es un conjunto de especificaciones que Microsoft ha suministrado para ayudar a los fabricantes de los compiladores. Estas especificaciones fijan el grupo mínimo de características que un lenguaje .net debe tener.

215. *¿Qué es una excepcion?*

Una excepción es cuando una app entra dentro de alguna condición que no puede resolver, esto hace que la app se cierre o genere errores.

216. *¿Qué se coloca en el bloque "Catch"?*

La excepción que necesitamos capturar.

217. *¿Cómo construiría un objeto del tipo "Exception" personalizado?*

Se debe extender la clase Exception para personalizar una excepción.

218. *¿Qué ocurre si en el bloque de código donde se produce la excepción el error no esta siendo tratado?*

Si existe la excepción genérica, el error se atrapará por más que no haya sido identificado. Caso contrario, se interrumpe la ejecución y el programa falla y finaliza.

219. *¿Cuál es el objeto de mayor jerarquía para el manejo de excepciones?*

El objeto de mayor jerarquía y por lo tanto menos especifico es el objeto de tipo Exceptión

220. *¿En que namespace se encuentra la clase Exception?*

En System.

221. *¿Cuáles son las dos clases genéricas más importantes definidas en el framework ademas de Exception?*

- System.System.Exception (la mayoría los obj. Exception definidos en .NET Framework heredan de aquí)
- System.AplicationException (los obj de excepción personalizados y específicos de cada aplicación heredan de aquí)

222. *¿Qué instrucción se utiliza para poner en práctica el control e interceptar las excepciones?*

Se utiliza el Catch

223.
¿Dónde se coloca el código protegido contra excepciones si se iniciara una excepción?

Se coloca en el Finally

224.
¿Qué tipo de excepción se utiliza para interceptar un error de división por cero?

DivideByZeroException

225.¿Qué tipo de excepción se utiliza para interceptar una DLL que tiene proble mas al ser cargada?

FileLoadException

226. ¿Qué colocaría dentro de una clausula "Catch" para especificar una condición adicional que el bloque "Catch" deberá evaluar como verdadero para que se seleccionada?

Colocaría la palabra reservada When para evaluar una condición de ingreso.

227.¿Si se desea colocar código de limpieza y liberación de recursos para que se ejecute cuando una excepción se produzca, dónde lo colocaría?

Dentro de la misma excepción.

228. *¿Qué instrucción se utiliza para provocar un error y que el mismo se adapte al mecanismo de control de excepciones?*

Se utiliza la instrucción Throw, que lanza la excepción para que la capture el segmento de código en donde se encuentra contenido.

229. *¿Escriba el código que permitiría provocar la excepción del tipo "ArgumenException"?*

230. *¿Como construiría un objeto del tipo "Exception" personalizado?*

231. *¿Cómo armaría un "Catch" personalizado para que se ejecute cuando se de la excepción "ClienteNoExisteException"?*

```
Try{

cargarCliente(cliente);

}

Catch (ClienteNoExisteException ex){

MessageBox.Show(ex.Message);

}

Private void cargarCliente(cliente)
{
        If (cliente.id=0)
        {
                Throw new ClienteNoExisteException("Cliente invalido");
        }
 }
```

232. *¿Dónde se encuentran las instancias de los objetos administrados por el GC?*

Las instanciad de los objetos se encuentran en el Heap.

233. *¿Cuáles son los dos métodos más notorios que deben implementar las clases para trabajar correctamente con la recolección de elementos no utilizados y matar las instancias administradas y no administradas?*

Los métodos que se debe implementar son Finalize y Dispose.

234. *¿De dónde heredan las clases el método Finalize?*

Heredan de System.Object.

235. *¿Cuál es la firma que implementa el método "Finalize"?*

236. *¿Qué método se utiliza para que el GC recolecte los elementos no utilizados?*

GC.Collect()

237. *¿Qué método se utiliza para suspender el subproceso actual hasta que el subproceso que se está procesando en la cola de finalizadores vacíe dicha cola?*

waitForPendingFinalizers

238. *¿Qué método se ejecuta en los objetos cuando se ejecuta el método collect del GC?*

Se ejecuta el método Finalize().

239. *¿Qué método debería exponer una clase bien diseñada teniendo en consideración que no posee destructor?*

Se debería exponer el método Dispose, para esto se debe implementar la interfaz Idisposable

240. *¿Cómo obtengo el método "Dispose"?*

Se obtiene implementando la interfaz IDisposable.

241. *¿Qué se programa en el método "Dispose"?*

En el método Dispose se programa el código para liberar recursos utilizados por el objeto.

242. *¿Se pueden combinar el uso de "Dispose" y "Finalize"?*

Si se puede combinar y resulta buena práctica hacerlo porque el código suele ser el mismo.

243. *¿A qué se denomina "Resurrección de Objetos"?*

La resurrección de objetos es un mecanismo que se utiliza para recuperar objetos que han sido eliminados y limpiados por el garbage collector.

244. *¿A qué se denomina "generación" en el contexto de la recolección de elementos no utilizados?*

El recolector de elementos no utilizados cuenta con un método para determinar la edad de un objeto. Cada objeto mantiene un contador que dice el número de recolecciones de elementos no utilizados a las que el objeto ha

sobrevivido. El valor de este contador es la generación del objeto. Cuando más elevado sea este número, menor será la posibilidad de que dicho objeto sea eliminado durante la siguiente recolección de elementos no utilizados.

245. *¿Qué valores puede adoptar la "Generación" de un objeto?*

Hay sólo tres valores distintos de generación. El valor de la generación de un objeto que nunca ha sufrido una recolección de elementos no utilizados es 0; si el objeto sobrevive a una recolección de elementos no utilizados su generación vale 1; si sobrevive a una segunda recolección de elementos no utilizados, su generación valdrá 2. Cualquier posterior recolección de elementos no utilizados hace que el contador de generación siga valiendo 2 (o se destruye el objeto).

246. *¿Cómo se puede obtener el numero de veces que se ha producido la recolección de elementos no utilizados para la generación de objetos especificada?*

GC.Collect(x) en x se pasa el valor mencionado en el punto anterior que se quiere verificar.

247. *¿Cómo se obtiene el número de generación actual de un objeto?*

GC.GetGeneration()

248. *¿Cómo se puede recuperar el número de bytes que se considera que están asignados en la actualidad?*

GC.GetTotalMemory()

249. *¿Qué utiliza para convertir un objeto en "no" válido para la recolección de elementos no utilizados desde el principio de la rutina actual hasta el momento en que se llamó a este método?*

GC.SupressFinalize()

250. *¿Cómo se solicita que el sistema no llame al finalizador del objeto especificado?*

GC.SupressFinalize()

251. *¿Cómo se solicita que el sistema llame al finalizador del objeto especificado, para el que previamente se ha llamado a "SuppressFinalize"?*

GC.WaitForPendingFinalizers()

252. *¿Cómo se obtiene el número máximo de generaciones que el sistema admite en la actualidad?*

GC.GetGeneration()

UNIDAD 4

1. *¿Para que se utilizan las interfaces?*

La interfaz se utiliza para obligar a definir a las clases que la implementan un comportamiento particular para cada una de dichas clases.

2. *¿Cómo se implementa una interfaz?*

Solo se necesita colocarlas luego de los : a continuación del nombre de la clase.

3. *¿Se pueden heredar las interfaces?*

Si, las interfaces pueden heredarse.

4. *¿Se puede implementar un tipo de polimorfismo peculiar por medio de interfaces?*

Si se implementa un tipo de polimorfismo en el que se define el contrato y las clases que implementan la interfaz definen como realizar la implementación de sus métodos.

5. *¿Para qué se utiliza la interfaz IComparable?*

Se utiliza para ordenar únicamente por un solo criterio de ordenamiento. Al utilizar la interfaz IComparable se podrá ordenar los objetos de una lista utilizando el método Array.Sort, para ello se tendrá que escribir el método compareTo, que recibe un objeto y devuelve -1,0 o 1, dependiendo de que el objeto actual sea menor que, igual a o mayor que el objeto que se ha pasado como argumento.

6. *¿Para qué se utiliza la interfaz IComparer?*

La mayoría de los objetos del mundo real se puede comparar y ordenar atendiendo al contenido de diferentes campos o combinaciones de campos, en este caso, se podrá utilizar una variedad del método Array.Sort que acepta una interfaz IComparer como segundo argumento. La interfaz IComparer expone únicamente un método, Compare, que recibe dos referencias a objetos y que devuelve -1,0,1 dependiendo de que el primer objeto sea menor que, igual a, o mayor que, el segundo objeto.

7. *¿Para qué se utiliza la interfaz ICloneable?*

Es una interfaz que provee las herramientas para realizar la clonación de objetos en distintos niveles.

8. *¿Para qué se utiliza la interfaz IEnumerable?*

Enumera elementos de una clase desde afuera de la misma, entregando una lista de elementos y no la colección completa. Obliga a implementar getEnumerator() que devuelve un Enumerador. Recupera todos los elementos de un enumerador. Esta interfaz deberá devolver un objeto que permita el empleo de la interfaz IEnumerator.

9. *¿Para qué se utiliza la interfaz IEnumerator?*

Para crear un numerador personalizado. Posee los siguientes métodos:

- Funcion MoveNext: Este método se llama en cada una de las iteraciones ForEach y debe devolver True si existe un nuevo valor o False si no existen más elementos.
- Propiedad de solo lectura Current: Devuelve el valor actual.
- Procedimiento Reset: Este método redefine el puntero interno para que el siguiente valor que se devuelva sea el primero de una nueva serie.

Estas dos interfaces funcionan conjuntamente para proporcionar compatibilidad For Each para que las clases se comporten como una clase colección ante sus clientes.

10. *¿Qué es un delegado?*

Un delegado es un tipo que representa referencias a métodos con una lista de parámetros determinada y un tipo de valor devuelto. Cuando se crea una instancia de un delegado, puede asociar su instancia a cualquier método mediante una signatura compatible y un tipo de valor devuelto.

11. *¿A qué elementos se les puede delegar?*

Los delegados permiten pasar los métodos como parámetros.

12. *¿Qué se puede delegar?*

Se pueden delegar eventos y operaciones asincronicas

13. *¿Cómo construiría un delegado?*

public void delegate <Nombre del delegado>(<firma>)

14. *¿Cómo implementaría in procedimiento de devolución de llamadas?*

```
class CLIENTE
   {
      public delegate void delegateLogMessage(string mensaje);

      public event delegateLogMessage eventMessage;

      public void GenerarMensaje()
      {
         eventMessage("Este es un mensaje del evento");
      }
   }
```

15. *¿Para qué sirve la multidifusión de delegados?*

Para enviar una llamada a más de un procedimiento.

UNIDAD 5

16. *¿Qué son y para que se usan los tipos genéricos?*

Los genéricos son esencialmente una "plantilla de código" que permite a los desarrolladores definir estructuras de datos con seguridad de tipos sin confirmar un tipo de datos real.

17. *¿A partir de que versión de C# se pueden utilizar?*

A partir de la versión 2.0 del framework.

18. *Enumere las ventajas de utilizar genéricos.*

- Definen el tipo de dato que va a contener cada instancia de la lista para que el tipo de dato quede definido en tiempo de compilación.
- En tiempo de ejecución se conoce el tipo de estructura que se está utilizando, así se almacenan en memoria de forma más eficaz.
- Al conocer el tipo en tiempo de ejecución también se mejora la depuración, ya que podemos ver en detalle cada elemento.

19. *¿A qué elementos se le pueden aplicar genericos?*

Diccionarios, listas, clases, interfaces, eventos y delegados.

20. *¿Cuáles son los usos más comunes de los genéricos?*

En las interfaces genéricas como IEnumerate, a través de LinQ. Al generar listas con List<>, cuando creamos delegados con EventHandles<>, o eventos con EventArgs, cuando hacemos pilas y colas como Queue<> y Stack<>. Al generar Diccionarios con Dictionary<>, etc.

21. *¿Qué aspectos trascendentes se deben considerar al crear clases genéricas?*

.

22. *¿Cuál es la diferencia entre heredar de una clase genérica abierta y una clase genérica cerrada?*

Las clases no genéricas, en otras palabras, concretas, pueden heredar de clases base construidas cerradas, pero no desde clases construidas abiertas ni desde parámetros de tipo porque no hay ninguna manera en tiempo de ejecución para que el código de cliente proporcione el argumento de tipo necesario para crear instancias de la clase base. Las clases no genéricas, en otras palabras, concretas, pueden heredar de clases base construidas cerradas, pero no desde clases construidas abiertas ni desde parámetros de tipo porque no hay ninguna manera en tiempo de ejecución para que el código de cliente proporcione el argumento de tipo necesario para crear instancias de la clase base. Tipos construidos cerrados y construidos abiertos pueden usarse como parámetros de método.

23. *¿Por qué es importante colocar restricciones en los parámetros de tipo?*

Las restricciones informan al compilador sobre las capacidades que debe tener un argumento de tipo. Sin restricciones, el argumento de tipo puede ser cualquier tipo.

24. *¿Cuáles son los tipos de restricciones que se le pueden colocar a un parámetro de tipo?*

Se puede agregar restricciones no administradas, restricciones NotNull, .

25. *Ejemplifique como crearía un método genérico con un parámetro de tipo*

Es similar a un puntero a una función de C en el sentido que permite llamar a un procedimiento utilizando un puntero que apunte al propio procedimiento.

26. *¿Qué es LinQ?*

LinQ es una instrucción que se utiliza para poder filtrar resultados de una lista mediante una consulta o query con los filtros deseados.

27. *¿Qué orígenes de datos se pueden consultar con LinQ?*

28. *¿Qué especifica la consulta en una estructura de LinQ?*

Las 3 partes básicas de una consulta LinQ son:

- From: donde especificamos el origen de datos.
- Where: donde establecemos las condiciones que vamos a imponer para traer los datos
- Select: donde definimos los datos que queremos traer.

28. *¿Mencione al menos 3 clausulas (las más importantes) que se usan en una expresión de consulta LinQ?*

29. *¿?*

SELECT, FROM, WHERE

31. *¿Explique que hace cada clausula enumerada en la pregunta anterior?*

- From: donde especificamos el origen de datos.
- Where: donde establecemos las condiciones que vamos a imponer para traer los datos
- Select: donde definimos los datos que queremos traer

32. *¿Qué métodos implementa IEnumerable?*

Count, distinct, First, FirstOrDefault, group by, sort, take

33. *¿Qué utilizaría para ordenar en una expresión LinQ?*

Order by

34.

¿Qué utilizaría para lograr una unión entre dos origenes de datos en una expres ión LinQ?

union

35. *¿Cómo se pueden generar nuevos tipos utilizando LinQ?*

36. *¿Qué es una expresión Lambda?*

37.

Ejemplifique cómo se puede utilizar una expresión lambda para realizar una co nsulta.

38. ¿Qué debo realizar para crear una expresión lambda?

39.

¿Indique como puede declarar un delegado utilizndo Func y que significa cada e lemento utilizado?

40.

Crear un delegado utilizando Func que posea tres parámetros de entrada y uno de salida. Los parámetros de entrada son: el primero int, el segundo double y el tercero bool. El parámetro de retorno es de tipo bool

UNIDAD 6

41. *¿Qué características posee un esquema cliente - servidor?*

Se tiene una máquina cliente, que requiere un servicio de una máquina servidor, y éste realiza la función para la que está programado (nótese que no tienen que tratarse de máquinas diferentes; es decir, una computadora por sí sola puede ser ambos cliente y servidor dependiendo del software de configuración). Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma

42. *¿Qué significa pasar información batch?*

Implica procesar varias transacciones al mismo tiempo, y no se dispone inmediatamente de los resultados del resto de transacciones cuando comienza cada una de ellas para un mejor funcionamiento de un sistema.

43. *¿Qué significa pasar información on line?*

El concepto se utiliza para nombrar a algo que está conectado o a alguien que está haciendo uso de una red (generalmente, Internet). Se dice que la

información está online o en línea, cuando se encuentra disponible a través de Internet.

44. *¿Qué es un protocolo?*

Es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red por medio de intercambio de mensajes. Éste es una regla o estándar que controla o permite la comunicación en su forma más simple, puede ser definido como las reglas que dominan la sintaxis, semántica y sincronización de la comunicación. Los protocolos pueden ser implementados por hardware, software, o una combinación de ambos. A su más bajo nivel, éste define el comportamiento de una conexión de hardware.

45. *¿Qué protocolo usa una red de área local?*

TCP/IP

46. *¿Qué protocolo usa internet?*

IPv4 // IPv6 (futuro cercano)

47. *¿Qué hace el protocolo IP?*

IP es el protocolo encargado del transporte de paquetes desde el origen hasta el destino en una comunicación. Es un protocolo que no garantiza la fiabilidad aunque trata de hacer todo lo posible para que los paquetes lleguen al destino. IP se basa en el encaminamiento que se produce entre dispositivos de la capa 3 (OSI) y en los identificadores únicos (direcciones IP) que asigna a cada dispositivo para que pueda comunicarse. Cada dispositivo de capa 3 realiza el proceso de encaminamiento en base a su tabla de rutas, que le indica el camino más adecuado para llegar a cada destino.

48. *¿Qué ventajas tiene distribuir procesos?*

Los sistemas distribuidos están basados en las ideas básicas de transparencia, eficiencia, flexibilidad, escalabilidad y fiabilidad. Por lo tanto los sistemas distribuidos han de cumplir en su diseño el compromiso de que todos los puntos anteriores sean solucionados de manera aceptable.

49. *¿Qué ventajas tiene distribuir almacenamientos?*

Se conoce como sistema de almacenamiento distribuido a todo aquel que permite almacenar ficheros online. La principal característica es poder guardar archivos (documentos, imágenes, vídeos...) en la red. Una de las cosas que caracterizan al almacenamiento distribuido es el gran rango de aplicaciones que tiene. Las tres más importantes son:

- Copias de seguridad de los archivos.
- Compartir archivos en red.

50. *¿Qué es un socket?*

Un Socket es una relación entre un puerto de un equipo y el puerto de otro equipo. Los puertos son básicamente, una entrada/salida de información. Estos se encuentran identificados por un número entero y muchos se encuentran reservados para determinadas tareas, por ejemplo: el puerto 80 es para un servidor web.

51. *¿Qué característica posee un socket sincrónico?*

Quien envía permanece bloqueado esperando a que llegue una respuesta del receptor antes de realizar cualquier otro ejercicio

52. *¿Qué característica posee un socket asincrónico?*

Quien envía continúa con su ejecución inmediatamente después de enviar el mensaje al receptor.

54. *¿Para qué se utilizan los puertos de la pc?*

Un puerto es una forma genérica de denominar a una interfaz a través de la cual los diferentes tipos de datos se pueden enviar y recibir. Dicha interfaz puede ser de tipo físico, o puede ser a nivel de software (por ejemplo, los puertos que permiten la transmisión de datos entre diferentes ordenadores)

55. *¿Qué puertos posee una PC?*

Paralelo – Serie – USB

56. *¿Cómo funciona el puerto paralelo?*

En un esquema de transmisión de datos en paralelo un dispositivo envía datos a otro a una tasa de n número de bits a través de n número de cables a un tiempo. Sería fácil pensar que un sistema en paralelo es n veces más rápido que un sistema en serie, sin embargo esto no se cumple, básicamente el impedimento principal es el tipo de cable que se utiliza para interconectar los equipos.

57. *¿Cómo funciona el puerto serie?*

En un esquema de transmisión de datos en serie un dispositivo envía datos a otro a razón de un bit a la vez a través de un cable.

58. *¿Cómo funciona el puerto USB?*

Un buen punto de partida para abordar este tema es el cableado del bus. Cada cable USB contiene, a su vez, 4 cables en su interior. Dos de ellos están dedicados a la alimentación (5v) y la referencia de tensión (masa). La corriente máxima que el bus puede proporcionar es de 500 mA a 5 voltios de tensión.

Los dos cables restantes forman un par trenzado, que transporta la información intercambiada entre dispositivos, en formato serie. Tras su encendido, el dispositivo anfitrión -el PC- se comunica con todos los dispositivos conectados al bus USB, asignando una dirección única a cada uno de ellos (este proceso recibe el nombre de "enumeración"). Además, la PC consulta qué modo de transferencia desea emplear cada dispositivo: por interrupciones, por bloques o en modo isócrono.

La transferencia por interrupciones la emplean los dispositivos más lentos, que envían información con poca frecuencia (por ejemplo teclados, ratones, etc.). La transferencia por bloques se utiliza con dispositivos que mueven grandes paquetes de información en cada transferencia. Un ejemplo son las impresoras. Finalmente, la transferencia isócrona se emplea cuando se requiere un flujo de datos constante y en tiempo real, sin aplicar detección ni corrección de errores. Un ejemplo es el envío de sonido a altavoces USB. Como se puede intuir, el modo isócrono consume un ancho de banda significativo. Por ello el PC impide este tipo de transferencia cuando el ancho de banda consumido supera el 90% del ancho de banda disponible.

Para la temporización, el bus USB divide el ancho de banda en porciones, controladas por el PC. Cada porción mueve 1.500 bytes, y se inicia cada milisegundo. Ante todo, el PC asigna ancho de banda a los dispositivos que emplean transferencias isócronas y por interrupciones, garantizando el ancho de banda necesario. Las transferencias por bloques emplean el espacio restante, quedando en última prioridad.

59. ¿Qué es la domótica?

La domótica es la integración de tecnología en el diseño inteligente o automatizado de un recinto (Casa, Apartamento, Casas campestres, fincas, lugar de trabajo etc…) con funciones de información, entretenimiento, gestión energética, seguridad y búsqueda de soluciones a la medida y aplicaciones según sus necesidades. Entonces, se trata de una ciencia que estudia la aplicación de la informática y las comunicaciones al hogar, con el fin de conseguir una -casa inteligente-. La domótica pretende, por ejemplo, que las luces, calefacción, etc., se regulen automáticamente en función de las condiciones exteriores, consiguiendo de paso un considerable ahorro energético.